# Introduction to mrgsolve : Hands on tutorial

Kyle Baron and Ana Ruiz

Metrum Research Group and Gilead Sciences

Red Iberoamericana de Farmacometria 2022
September 30, 2022

Red Iberoamericana de Farmacometría

# Presentation Overview

# The Brain

## Kyle Baron, Science Advisor, Principal Scientist II at Metrum Research Group

# What is R?

## Introduction

- R is a language and environment for statistical computing and graphics.
- R provides a wide variety of statistical and graphical techniques, and is highly extensible. R provides an Open Source route to participation in that activity.
- Well-designed publication-quality plots can be produced while user retains full control.
- R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS.
- More information can be found at: hhtp://r-project.org/

# RStudio

## To posit means to put forth an idea for discussion

- **RStudio Workbench** is an integrated development environment for R and Python, with a console, syntax-highlighting editor that supports direct code execution, history, debugging and workspace management.
- **RStudio Cloud** is a lightweight, cloud-based solution that allows anyone to do, share, teach and learn data science online.
- **RStudio Connect** allows you to share data products across your organization (Shiny applications, R Markdown reports, Jupyter Notebooks, and more).
- **RStudio Package Manager** will allow you to control, organize, and govern your use of R packages.

# Command Line

## R packages and more

## Installing packages

```
1 install.packages("tidyverse")
2 library(tidyverse)
3 update.packages("tidyverse")
```

## Getting help

```
1 help("mutate")
2 ?mutate
3 example(mutate)
```

# Command Line

## Paths

## Paths

```
.GlobalEnv or globalenv()
sessionInfo()
.libPaths()
setwd("/cloud/project/script")
getwd()
```

# Command Line

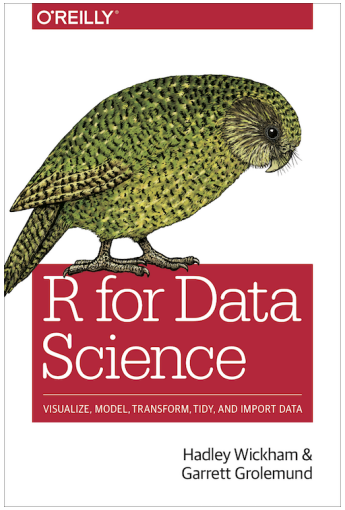
## Open/Save objects

## Open/Save objects

```
1 read.csv(file="example.csv",na=".")
2 read.table("test.csv", sep=",", header=TRUE, skip=1,
      as.is=TRUE)
3 write.csv(d,"test.csv",quote=F,na=".",row.names = F)
```

## Learning Tidy

- The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.
- Following three rules makes a dataset tidy: variables are in columns, observations are in rows, and values are in cells.
- "R for Data Science". Read it online at ▸ R for Data Science



# Data

## Basic Types

### Basic Types

R has six basic ('atomic') data types: logical, numeric, integer, complex, string (or character) and raw. The modes and storage modes for the different vector types are listed in the following table.

- **logical**: logical data (TRUE or FALSE). Also known as boolean data type, it can only take 2 values
- **numeric**: all real numbers with or without decimal values
- **integer**: real values without decimal points
- **complex**: purely imaginary values (3+2i)
- **character**: character strings values
- **raw**: (unususal) specifies values as raw bytes

## Data Objects

Vectors and factors are built using atomic data types. Vectors can be thought of as contiguous cells containing data. Cells are accessed through indexing operations such as x[5].

- **vector**: a list of atomic values
- **factor**: variables that can take on a limited number of different values. This relates to the concept of levels, where the level of a factor is basically the number of distinct elements

# Piping

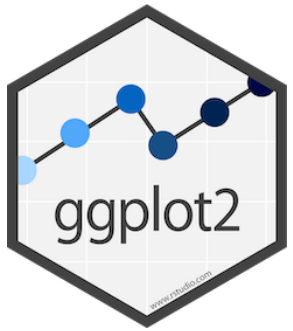
## magrittr

**magrittr**

- Pipes are an extremely useful tool from the magrittr package 1 that allow you to express a sequence of multiple operations. They can greatly simplify your code and make your operations more intuitive. However they are not the only way to write your code and combine multiple operations.
- **open R file datamanipulation**

## Data visualization

- ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphics. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.
- The Data Visualisation and Graphics for communication chapters in R for Data Science will get you up to speed with the essentials of ggplot2 as quickly as possible.
- cheat sheet included under Documents (data-visualization-2.1.pdf)
- **open R file ggplot2**



# Introduction

## mrgsolve

- mrgsolve is an R package for simulation from hierarchical, ordinary differential equation (ODE) based models typically employed in drug development. mrgsolve has been used for a wide variety of model applications, including pharamacokinetics (PK), pharmacokinetics/pharmacodynamics (PK/PD), physiologically-based pharmacokinetic (PBPK) modeling, and quantitative systems pharmacology.
- mrgsolve is is free, open-source software.

# Online Resources

## Online Resources

- ▸ mrgsolve, this website includes:
  - User Guide
  - Package documentation
  - R documentation
  - Doxygen documentation
  - Vignettes
  - Demos
- ▸ mrgsolve github A github repository of short, focused, how-to vignettes

# Model Specification

## Model Specification

We are going to learn how to write a mrgsolve model using a separate file and source it into an R script. The following code blocks formualted either with a dollar-sign placed at the strat of the block name or the block name is betweenn squared brackets

- $PROB or [PROB]
- $GLOBAL or [GLOBAL]
- $MAIN or [MAIN] (AKA $PK)
- $PKMODEL or [PKMODEL]
- $PARAM or [PARAM]
- $CMT or [CMT]
- $ODE or [ODE] (AKA $DES)
- $THETA or [THETA]
- $OMEGA or [OMEGA]
- $SIGMA or [SIGMA]
- $SET or [SET]
- $CAPTURE or [CAPTURE]
- $TABLE or [TABLE] (AKA $ERROR)

# $PROB

## $PROB

Use this block to make notes about the model. There are no restrictions on the text that gets entered here. mrgsolve does not routinely process the text in any way, except when rendering the model as a document

```
$PROB
# Model: `pk1cmt`
  - One-compartment PK model
      - Dual first-order absorption
      - Optional nonlinear clearance from `CENT`
  - Source: `mrgsolve` internal library
  - Date: `r Sys.Date()`
  - Version: `r packageVersion("mrgsolve")`
```

# $GLOBAL

## $GLOBAL

The $GLOBAL block is for writing C++ .

- Often used for preprocessor directives using #define
- Declare global variables

```
[ GLOBAL ]
#define CP (CENT/VC)

[ global ]
double TVCL, TVV2, TVQ = 0, TVV3 = 0;

$GLOBAL
bool cure = false;
```

# $MAIN

## $MAIN or $PK

This code block has two main purposes:

- Derive new algebraic relationships between parameters, random, effects and other derived variables
- Set the initial conditions for model compartments

```
[MAIN]
double TVCL      = THETA1;
double CL_AGE    = THETA5;
double LOGTWT = 0.75*log((WT/70.0));
double LOGTAGE = log((AGE/35.0));
double CL =  exp(log(TVCL) + CL_AGE * LOGTAGE +
    LOGTWT);
```

# $PKMODEL

## $PKMODEL

This code block implements a one- or two-compartment PK model where the system is calculated by algebraic equations, not ODEs

```
[ CMT ] GUT CENT PERIPH
[ PKMODEL ] ncmt=2, depot=TRUE
```

# $PARAM

## $PARAM

Define the parameter list in the current model. Parameters are names associated with values that can be used throughout the model. A value must be given for every parameter name. Names (and numbers) of parameters must be set at the time the model is compiled, but parameter values may be updated without re-compiling the model.

```
$PARAM
TVKA = 0.5, TVCL = 1, TVV = 24
$PARAM
@covariates
WT = 70
```

# $CMT and $INIT

## $CMT and $INIT

Declare the names of all compartments in the model.

- For $CMT give the names of compartments; initial values are assumed to be 0
- For $INIT give the name and initial value for all compartments

Note that both $CMT and $INIT declare compartments, so any compartment name should get declared in either $CMT or $INIT, but never both.

```
[ CMT ] GUT CENT RESPONSE
[ INIT ] GUT  = 0, CENT = 0, RESPONSE = 25
[ CMT ] @annotated
GUT      : Dosing compartment (mg)
CENT     : Central PK compartment (mg)
RESPONSE : Response
```

## $ODE or $DES

Use $ODE to define model differential equations. For all compartments assign the value of the differential equation to dxdt_CMT where CMT is the name of the compartment.

```
[ CMT ] GUT CENT
[ ODE ]
dxdt_GUT = -KA*GUT;
dxdt_CENT = KA*GUT - KE*CENT;
```

# $THETA

$THETA

Use this code block as an efficient way to add to the parameter list where names are determined by a prefix and a number.

```
[ THETA ]
0.1 0.2 0.3
which is equivalent to:
$PARAM THETA1 = 0.1, THETA2 = 0.2, THETA3 = 0.3
```

# $OMEGA

## $OMEGA

Use this block to enter variance/covariance matrices for subject-level random effects drawn from multivariate normal distribution

```
1 the default is the diagonal
2 $OMEGA
3 1 2 3
4 for a block:
5 $OMEGA @block
6 0.1 0.02 0.3
```

# $SIGMA

## $SIGMA

Use this block to enter variance/covariance matrices for within-subject random effects drawn from multivariate normal distribution

```
1 $SIGMA
2 1 2
3 for a block:
4 $SIGMA @block
5 0.1 0.02 0.3
```

## $SET

Use this code block to set different options for the simulation

```
1 [ SET ] end = 240, delta = 0.5
```

# $CAPTURE

## $CAPTURE

- This is a block to identify variables that should be captured in the simulated output
- Capture may be used to declare variables in $MAIN and $TABLE
- Captured variables can be renamed by providing a newname = oldname specification

```
1 [ PARAM ] A = 1, B = 2
2 [ MAIN ]
3 double C = 3;
4 bool yes = true;
5 [ CAPTURE ] A B C yes
6 $TABLE
7 capture DV = (CENT/VC);
8 $CAPTURE WEIGHT = WT TVCL = THETA2 CL  ETA(1)
```

# $TABLE or $ERROR

## $TABLE or $ERROR

Use this code block to interact with parameters, compartment values, and other user-defined variables after the system advances to the next time.

```
[ TABLE ]
double CP = CENT/VC;
$SIGMA
0.0639
$TABLE
capture IPRED = CENT/(V2/1000);
capture DV = IPRED*(1+EPS(1));
```

# A Basic Simulation

## mread()

### Use mread() to read, compile and load a model

```
1 modlib(list=TRUE)
2 mod <- mread("pk1", modlib())
3 mod
```

- pk1cmt: one compartment pk model absorption with optional non linear elimination using ODEs
- pk2cmt: two compartment pk model with dual absorption with optional non linear elimination using ODEs
- pk3cmt: three compartment pk model with dual absorption with optional non linear elimination using ODEs
- pk1: one compartment pk model in closed-form
- pk2: two compartment pk model in closed-form
- popex: a simple population pk model in closed-form

# A Basic Simulation

## mread()

Use mread() to read, compile and load a model

```
1 modlib(list=TRUE)
2 mod <- mread("pk1", modlib())
3 see(mod)
```

- pk1cmt: one compartment pk model absorption with optional non linear elimination using ODEs
- pk2cmt: two compartment pk model with dual absorption with optional non linear elimination using ODEs
- pk3cmt: three compartment pk model with dual absorption with optional non linear elimination using ODEs
- pk1: one compartment pk model in closed-form
- pk2: two compartment pk model in closed-form
- popex: a simple population pk model in closed-form

# A Basic Simulation

## Components

```
1 model %>% intervention() %>% simulate() %>% post_
      process()
2 mod %>% ev(amt = 100, ii = 24, addl = 3) %>% mrgsim()
       %>% plot()
```

The basic elements are:

- mod: model object
- ev(amt=100, ...): an intevention
- mrgsim(): the simulation
- plot(): plots the simulation

# A Basic Simulation

## Intervention

```
1 ev()
2 ev_seq()
3 ev_rep()
4 seq()
5 expand.ev()
```

### defaults: time, evid, cmt

- time: event time
- evid: event id
- cmt: event compartment
- amt: dose amount
- ii: inter-dose interval
- addl: additional doses to administer
- total: total number of doses to administer
- rate: infusion rate
- tinf: infusion duration
- ss: if ss=1 advances to steady-state
- ID: subect ID

# A Basic Simulation

## ev(...)

- Bolus dosing (evid 1, rate==0)
- zero order infusion (evid 1, rate >0)
- other type of event (evid 2)
- compartment reset (evid 3)
- compartment reset and dose (evid 4)
- replace the amount in a specific compartment (evid 8)

# A Basic Simulation

## Types of simulation

- mod + ev + mrgsim
- mod + idata_set + ev + mrgsim
  - idata_set(): takes in individual-level data
  - ID-one per row
  - Typically parameters are in columns
  - idata_set object and mod are connected via parameters
- mod + data_set + mrgsim
  - data_set(): is the dosing equivalent to idata_set()
  - data_set() can carry parameters